

06-01-00 EK 483 548 47345

Box PATENT APPLICATION
Assistant Commissioner for Patents
Washington, D.C. 20231

DOCKET NUMBER: AUS990922US1

DATE: 5/31/00

Sir:

Transmitted herewith for filing is the Patent Application of:

Inventors: **Heather Maria Hinton and Mark Vandenwauver**
For: **AUTHENTICATION AND AUTHORIZATION PROTOCOL FOR SECURE WEB-BASED
ACCESS TO A PROTECTED RESOURCE**

Enclosed are:

- ☒ Patent Specification and ~~Declaration~~ **5**
- ☒ 3 sheets of drawing(s). (Informal)
- ☐ An assignment of the invention to International Business Machines Corporation (includes Recordation Form Cover Sheet).
- ☐ A certified copy of a ____ application.
- ☐ Information Disclosure Statement, PTO 1449 and copies of references.

The filing fee has been calculated as shown below:

For	Number Filed	Number Extra	Rate	Fee
Basic Fee				\$690.00
Total Claims	35-20	15	x \$18 =	270.00
Indep. Claims	5-3	2	x \$78 =	156.00
Multiple Dep.			x \$260 =	0
Claims Presented				
			TOTAL	\$1116.00

- ☒ Please charge my Deposit Account No. 090447 in the amount of \$1116.00. A duplicate copy of this sheet is enclosed.
- ☒ The Commissioner is hereby authorized to charge payment of the following fees associated with this communication or credit any overpayment to Deposit Account 090447. A duplicate copy of this sheet is enclosed.
 - ☒ Any additional filing fees required under 37 CFR 1.16.
 - ☒ Any patent application processing fees under 37 CFR 1.17.

Respectfully submitted,

By:

Jeffrey S. LaBaro
Jeffrey S. LaBaro
Registration No. 31,633
Intellectual Property Law Dept.
IBM Corporation
11400 Burnet Road, Zip 4054
Austin, Texas 78758

**AUTHENTICATION AND AUTHORIZATION PROTOCOL FOR SECURE
WEB-BASED ACCESS TO A PROTECTED RESOURCE**

BACKGROUND OF THE INVENTION

5 Technical Field

The present invention relates generally to techniques for enabling users on the Internet to securely access resources in various locations. More specifically, the invention relates to a web-based access control technique that uses a per-request client-generated token to authenticate that a request for access to a protected resource is properly bound with a given user identity.

Description of the Related Art

15 Information technology (IT) systems and the Internet have fueled the growth of the current global economy. While IT systems have significant benefits, at the same time they pose potential security threats from unauthorized third parties. Indeed, the lack of security
20 in modern IT systems has emerged as a threat to the integrity of global computer networks. To deal with this problem, IT systems provide a number of known services: data authentication, data confidentiality, entity authentication, and authorization, among others. Data
25 authentication typically consists of two sub-services, data integrity and data origin authentication. A data

integrity service is used to convince a receiver of given data that the data was not changed during transit. Data origin authentication proves to the receiver the identity of the real sender. Data confidentiality protects
5 against disclosure of data during transmission. Entity authentication provides the system with proof that a certain entity is who they claim to be. Authorization is the act of determining whether an authenticated entity has the right to execute an action. Authorization and
10 authentication thus are dual services. To be able to provide authorization, it is necessary to determine who the entity is (e.g., by entity authentication). Authorization, in general, consists of two separate stages: providing privileges (authorization credentials)
15 to a particular entity, and using these privileges in combination with access decision rules at the resource to determine if access should be granted to the entity.

It is becoming increasingly important to allow users to securely access resources in various locations. For
20 example, an employee of a company may need to access documents from a main office and also from a local office while located at home or at a customer's premises. A browser has become the tool of choice in such scenarios. Through the standard Hypertext Transfer Protocol (HTTP),
25 the browser can be used to access any HTTP-enabled server

(commonly called a Web Application Server (WAS)) and obtain access to the resource. Most browsers provide security through the Transport Layer Security (TLS) protocol. This protocol allows both the browser and the WAS to authenticate each other (i.e. to prove their identity to each other), and it also provides data protection (data integrity and data confidentiality) for data in transit between them. The strongest form of authentication provided by the TLS/SSL protocol is client- and server-side certificate authentication. Such authentication requires the client (the browser) and the server (the WAS) to each have a private/public cryptographic key pair, and associated certificates. Public key authentication maintains a binding between a user's identity and a public key that can only be unlocked by the associated private key, and these protocols are used to provide mutual authentication.

If the user at the client desires to access a URL on the server that can only be accessed by an authenticated and authorized user, however, there must be some process to determine authorization. SSL does not provide authorization (or other security services) to the Web Application Server. Therefore, although the server can be sure of the user's identity via authentication, it does not know the user's privileges.

One attempt to solve the authorization problem is to pass authentication information within a cookie. As is well-known, a cookie is a file that is set by a server to customize data to a particular user's web browser.

5 Cookies thus provide a degree of "state" to HTTP, which is otherwise a stateless protocol. When a user of a client machine visits a web server, the server may return a cookie to the user's browser. When a cookie is set as part of a HTTP transaction, it may include the path the
10 cookie is valid for, the cookie's name and value, and other optional attributes, such as the cookie's expiration date. By default, the browser automatically stores the cookie data, typically without giving the user the option or knowledge of it being done. Because the
15 cookie is stored, it is often referred to as "persistent." Later, when the user revisits the server, the cookie is sent with the request, thereby identifying the user to the server.

Thus, the typical persistent cookie set on a
20 client's browser identifies the user to the server. In prior art solutions, such as those provided commercially by enCommerce GetAccess™ and Netegrity SiteMinder™, authentication data is forwarded within a persistent cookie when the client browser issues a request for a
25 protected resource to the server that set that cookie.

Such an approach, however, is insecure because it enables an attacker to equate possession of the cookie with the user's authorization (i.e. a proof of identity) to access the protected resource. As a consequence, these prior art schemes are highly susceptible to replay attacks wherein one who acquires the identity cookie can simply assert it to gain access to the protected resource.

The present invention addresses this problem.

10

006372.00287:520006.01

BRIEF SUMMARY OF THE INVENTION

To ensure that a request for a protected resource is both authenticated and authorized, the present invention contemplates the client-side generation of a one-time only use authentication token that travels along with and protects an identity cookie that is forwarded to the server with the request. An authentication token of this type is used to validate the particular request to access a protected resource and to ensure that the user making the request, insofar as possible, is authorized. Preferably, the token is unique to a given request and unforgeable. It may include a time value that is used to protect against replay attacks that might otherwise occur due to network latencies. In an illustrative embodiment, the authentication token is protected by a key shared between the client and the server. Knowledge of this key implies authenticity of the client; thus, the key should be stored in a secure manner at the client. Preferably, the key resides in system memory only and is regenerated whenever a new identity cookie is produced.

When a user makes a request to access a protected resource identified by a URL, client-side code is used to generate the authentication token, which is then sent to the server along with an identity cookie that was set

by that server. In particular, the authentication token may be placed within the HTTP header and is forwarded to the server together with the identity cookie. The authenticated token is then used by the server to

5 authenticate that the request is properly tied to a given identity contained in the identity cookie. If the authentication token can be validated at the server, an access control decision is then executed to determine whether to invoke the request for the protected

10 resource. If the authentication token cannot be validated, an access denied request is returned to the requesting client.

The foregoing has outlined some of the more pertinent objects and features of the present invention.

15 These objects should be construed to be merely illustrative of some of the more prominent features and applications of the invention. Many other beneficial results can be obtained by applying the disclosed invention in a different manner or modifying the

20 invention as will be described. Accordingly, other objects and a fuller understanding of the invention may be had by referring to the following Detailed Description of the Preferred Embodiment.

25

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention and the advantages thereof, reference should be made to the following Detailed Description taken in
5 connection with the accompanying drawings in which:

Figure 1 illustrates a web-based environment in which the present invention may be implemented;

Figure 2 is a simplified illustration of how a requesting client communicates with a server to obtain
10 access to a protected resource on or associated with the server according to the present invention;

Figure 3 is a flowchart illustrating how a user obtains a authentication server-issued identity cookie for use in the authentication and authorization protocol
15 of the present invention;

Figure 4 is a flowchart illustrating the inventive protocol for enabling a user to obtain access to a protected resource using an authentication token that secures identity information in the identity cookie;

Figure 5 is a flowchart of a modified protocol of
20 the invention for use when the target server is different from the server that establishes the user's identity cookie; and

Figure 6 is a flowchart illustrating another modified protocol according to the invention for handling out-of-domain resource requests.

5

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Figure 1 illustrates a web-based environment in which the present invention may be implemented. In this environment, a user of a browser **102** at client **100**

5 desires to access a protected resource on web application server **104** in DNS domain **106**, or on web application server **108** in DNS domain **110**. A protected resource is a resource (an object, a document, a page, or the like) that is only retrieved if the requesting client browser

10 is both authenticated and authorized. Each DNS domain may have an associated authentication server **112**. Typically, once the user is authenticated by the authentication server, a cookie may be set and stored in a cookie cache in the browser. The requesting client may

15 make an intra-domain request or an inter-domain request for the protected resource. An intra-domain request means that the target resource is located on the same server that performs the authentication. An inter-domain request means that the target resource is located within

20 the same Internet domain but is on a different server than the authentication server which established the authentication. A cross-domain request means that the user wishes to access a protected resource that is outside its DNS domain.

Figure 2 illustrates how a requesting client may be used to request the protected resource according to the present invention. As illustrated, the user at a client workstation **200** seeks access over computer network **205** to a protected resource on a server **202** through the user's web browser **204**. As noted above, a protected resource is identified by a URL that can only be accessed by an authenticated and authorized user. The computer network **205** may be the Internet, an intranet, or other network.

Server may be a Web Application Server (WAS), a server application, a servlet process or the like. According to the invention, the client **200** receives an applet or equivalent code **206** from the server **202**. Applet **206** installs a plug-in or equivalent code module **208** that performs the client-side piece of the inventive functionality. The server-side piece of the functionality is provided by code module **210** that is resident in or accessible to the server. If the plug-in module **208** has been installed previously, the module is triggered by the server sending the client a custom MIME/type. The applet **206** preferably is signed so that it can get out of the Java sandbox. In this scenario, it is assumed that the user has been allocated a password in a secure way (preferably out-of-band), that this password is stored at the server side in a secure way, that the

applet **206** is signed with a key corresponding to a trusted certificate, that the browser **204** has been configured to execute signed applets from a trusted source, and that the server **202** needs to keep track of
5 the state of the system.

In a preferred embodiment, the applet is written in Java, although this is not a requirement. The plug-in may also be written in Java, but it is preferably native code. Because an embedded applet stops working when the
10 user switches to another web page, the applet cannot be used effectively to manage and perform cryptographic operations on the client. Therefore, preferably the signed Java applet is sent merely to install the plug-in.

As also seen in **Figure 2**, client browser **204** has a
15 cookie cache **214** in which an identity cookie **216** is stored. As is well-known, whenever the browser makes a request for a given URL at a server, the contents of the cookie cache are forwarded to the server. According to the present invention, a given URL request for a
20 protected resource also includes another piece of data **218**, referred to as an authentication token, which can be used at the server to authenticate whether the request is tied to a given identity in the identity cookie. If so, the server may initiate or invoke an access control
25 decision (ACD) to determine whether to afford access to

the protected resource. Preferably, the authentication token is a one-time only use token that is also time-dependent. The authentication token is generated by the browser plug-in or equivalent code and is preferably stored in system memory, not the cookie cache or other storage. Thus, the authentication token may be thought of as a non-persistent, unforgeable piece of data that facilitates authorization of the requestor when a given client request for a protected request is sent to a server. The token validates the identity cookie previously set by the server, as will be seen.

By way of brief background, the following notation is used to describe cryptographic operations in the present invention. $MAC(K)(m)$ refers to a Message Authentication Code calculated on a message m with a symmetric key K . S refers to a web server; C refers to a web client browser. K_{CS} is a long-term symmetric key shared between the client and the server. The term k_c refers to a short term symmetric key used to authenticate the client. The term k_{sc} refers to a short term symmetric key used to authenticate a request to access a protected resource at server S and to bind this request to client C . T refers to a timestamp, and N refers to a nonce (a non-repeating number) where N_s is a nonce generated by server S .

As will be seen, a user that issues a request to access a protected resource provides an identity cookie, which is a persistent cookie, that must first be provided or "set" by a given authentication server. The authentication server may or may not be the server that otherwise hosts the protected resource. In certain circumstances, however, a client may not possess such a cookie. One reason is that the client does not have the required plug-in that provides the inventive functionality; another reason is that the user has purged his or her cookie cache, where the persistent identity cookie is normally stored. Thus, by way of background, it may be necessary for the requesting client to obtain an authentication server-issued identity cookie as now described.

In particular, **Figure 3** is a flowchart illustrating how a user obtains an identity cookie and the component elements thereof according to the preferred embodiment of the invention. As will be seen, some of these steps occur on the client (through operation of the plug-in), while some of the steps occur on the server. The routine begins at step **300** with the client C initiating a request to visit a URL at server S. In this scenario, it is assumed that this request does not include an identity cookie. At step **302**, the server S receives the HTTP

request. A test is then performed at step **304** to determine whether the URL can only be accessed by an authenticated and authorized user. If the URL is available without restriction, the corresponding object,
5 document or page is returned at step **306**. If, however, the URL can only be accessed by an authenticated and authorized user, the routine continues at step **308**. At this step, the server sends the client a nonce (a non-repeating number) N_s and the signed applet that will
10 be used on the client to install and activate the required client-side code, namely, the plug-in. The routine then continues at step **310** with the client examining the MIME-type that corresponds to the plug-in. A test is then performed at step **312** to determine if the
15 MIME-type is found. If so, the routine branches to step **314** and calls the plug-in. If, however, the MIME-type is not found, the routine continues at step **316** with the applet installing and activating the plug-in.

At step **318**, which is also reached following step
20 **314**, the plug-in puts up a login window at the client. The user then enters his or her username and password at step **320**. Using this information, the plug-in then performs a number of calculations as follows. At step
322, the plug-in calculates the long term shared key K_{CS} ,
25 by applying a one-way function $h^{(1)}$ to the user's

password: $K_{CS} = h^{(1)}(PW_C)$. Then, at step **324**, the plug-in calculates the key through which the client authenticates to the server: $k_c = h^{(2)}(K_C, N_S, S)$. At step **326**, the plug-in calculates a Message Authentication Code (MAC) on the

5 nonce N_S , the identity of the server S that generated the nonce, a time T_0 , and the client's User-ID at S , UID_{CS} , as follows: $MAC(k_c)(N_S, S, T_0, UID_{CS})$. The client sends information about its time because there might be a clock skew in the system, i.e. a difference between the clock

10 running at the client and the clock running at the server. At step **328**, the client sends to the server the following token to enable the server to authenticate the client:

$$\{N_S, S, T_0, UID_{CS}, MAC(k_c)(N_S, S, T_0, UID_{CS})\}.$$

15 The routine then continues with the server S attempting to validate the token. At step **329**, the server calculates the symmetric session key, k_c , based on the user's password (stored at S), following the same steps as described above for the user's generation of k_c .

20 At step **330**, the server calculates the MAC itself. At step **332**, the server compares the MAC to the one it received in the token. A test is then performed at step **333** to determine if the MACs are equal. If not, the routine branches to step **335** to return an access denied

25 message to the client. This ends the protocol. If,

however, the MACs are equal, the server is convinced that the client is authentic. The routine then continues at step 336 to compare the client's time to the server's time. At step 338, the time difference (delta T or clock "skew") is stored in a data structure (representing the client's state) together with the client's authenticating key k_c .

The routine then continues at step 340 with the server obtaining an access control token (ACT) (sometimes known as an attribute certificate) for the client. In particular, the server may generate an ACT based on the client's UID_{cs} or it will call an external service (e.g., Policy Director) to provide one (e.g., a Policy Director EPAC). At step 342, the ACT is stored at the server in the client's state data structure and is used by the server S to make access control decisions on requests by the client C. At step 344, the server then generates the client's identity cookie for client C at server S. Preferably, the client's identity cookie contains the client's identifying information at S (e.g., UID_{cs} , which could be the user's DCE principal name), and a URL_{cs} that points to where other servers in the system can obtain up-to-date verification of a client's authenticator cookie (as will be described below). The identity cookie should be protected against tampering (thus providing

data authentication), for example, using a MAC or a digital signature. The identity cookie is a persistent cookie. A preferred format of the identity cookie is as follows, although any convenient structure may be used:

5 $IDC_{CS} = [UID_{CS}, S, URL_{CS}]$

The inclusion of the access control token in the identity token is optional:

$IDC_{CS} = [UID_{CS}, S, URL_{CS}], (ACT_{CS}).$

If it is included, it must be protected from theft,
10 disclosure and modification. The server identification, S, is included in the identity cookie following the practice given in M. Abadi and R. Needham, "Prudent Engineering Practice for Cryptographic Protocols", DEC SRC Research Report 125, June 1994.

15 At step 346, the identity cookie is returned from the server to the client. The client's browser then stores the identity cookie in the client's cookie cache at step 348 to complete the process.

The above-described scenario for generating a
20 persistent identity cookie may be used if the client already has the plug-in but has purged the cookie cache, the client has an inactive plug-in, or the client simply instigates a login. As can be seen, the identity cookie has additional data (besides the usual data, such as the
25 path the cookie is valid for, the cookie's name and

value, and other attributes, such as the cookie's expiration date) that will be used by the inventive protocol when the actual URL request is processed by the browser to attempt to fetch the desired resource.

5 In particular, the above-described protocol simply enables the user to obtain an identity cookie that may be used to facilitate the access request. **Figure 4** illustrates a protocol for carrying out an in-domain access request to the protected resource according to the present invention. As noted above, a technical advantage of the present invention is the client-side generation of an authentication token that is passed to the server along with the identity cookie to enable the server to authenticate that the request for access is tied to the identity contained in the identity cookie and to the requested resource. Only if the authentication token is valid does the server then allow the access control decision to proceed. The particular access control function (ACF) that is used to render the access control decision is not part of the present invention; any convenient ACF may be use.

10

15

20

The routine begins at step **400** with the client requesting a URL. At step **402**, client-side code in the plug-in recognizes that this URL is located in a domain

within the secure environment. At step **404**, the plug-in generates the authentication token preferably as follows:

$$ANC_c = \{N_s, S, T_0, UID_{CS}, MAC(k_c)(N_s, S, T_0, UID_{CS}, URL_{VISIT})\}.$$

As noted above, this token is used to authenticate the
5 identity cookie whenever the client tries to access the
protected resource. By including timestamp T_0 , the
authentication token (or "authenticator") is
time-dependent, although preferably the time period is
quite short and is merely used to compensate for network
10 latencies. By including the protected resource, URL_{VISIT} ,
the authentication token is bound to a request for that
resource. The plug-in preferably calculates a new
authentication token for every access the client requests
(even to the same URL) and this token preferably is not
15 placed in the client's cookie cache. In other words, the
authentication token is not persistent; rather, it is a
piece of information that is one-time use only.

The authentication token preferably uses the session
key k_{sc} , which can be obtained as:

$$20 \quad k_{sc} = h^{(3)}(k_c, S, IDC_{CS}),$$

where $h(3)$ is another one-way function. The one-way hash
functions $h(1)$, $h(2)$ and $h(3)$ may be the same, or
different. The key k_{sc} binds the authentication token to
the user identity inside the identity cookie and the
25 request to access the protected resource. In other

words, the authentication token is useful to authenticate that the request for the protected resource is bound to the identity contained within the identity cookie.

Moreover, because the timestamp T_0 is unknown to the
5 server, the information that is stored inside the authentication token is preferably:

$$N_s, S, T_0, \text{UID}_{CS}, \text{MAC}(k_{SC})(N_s, S, T_0, \text{UID}_{CS}, \text{URL}_{\text{VISIT}}).$$

The $\text{URL}_{\text{VISIT}}$ is included in the MAC so that the authentication token cannot be used to obtain access to a
10 resource other than the one requested by the client. The server's identity, S , is included in the MAC so that the authentication token is also bound to the server S .

Returning to the flowchart, the routine continues at step 406. At this step, the browser sends to the server
15 S the $\text{URL}_{\text{VISIT}}$, the identity cookie, and the authentication token as follows:

$$\text{IDC}_{CS} = [\text{UID}_C, S, \text{URL}_{CS}], (\text{ACT}_{CS})$$

$$\text{ANC}_C = N_s, S, T_0, \text{UID}_{CS}, \text{MAC}(k_{SC})(N_s, S, T_0, \text{UID}_{CS}, \text{URL}_{\text{VISIT}}).$$

Preferably, the authentication token is forwarded by the
20 browser after it is placed in the HTTP header of the client request by the plug-in. Any other convenient technique for transferring the token between the client browser and the server may be used. As is well-known, all cookies in the browser cookie cache that are relevant
25 to the server domain are forwarded with each HTTP

request, so the identity cookie is automatically forwarded. Processing then continues at the server S.

In particular, the routine continues at step **408** with the server looking up the delta T in the client state data structure. The routine then checks at step **410** to see whether the timestamp T_0 is within an allowable clock skew. As noted above, the clock skew may be set at a maximum by evaluating any expected network latencies. If the clock skew is not within an allowable limit, the access request is denied at step **412**. If the timestamp T_0 is within the allowable clock skew, the routine continues at step **414** with the server determining the identity of the requestor, UID_{cs} , from the identity cookie. At step **416**, the server then calculates k_{sc} in the same way as the client plug-in based on the value UID_{cs} pulled from the identity cookie. At step **418**, the server calculates the MAC and, at step **420**, verifies whether it is equal to the MAC received in the authentication cookie. A keyed hashing technique, such as described in H. Krawczyk, HMAC: Keyed-Hashing for Message Authentication, RFC 2104, February, 1997, may be used for this purpose.

If the outcome of the test at step **420** indicates that the MACs are not the same, the routine branches to step **422** and returns an access denied message to the

client to end the protocol run. If, however, the MAC generated is the same as the MAC in the authentication token, the routine continues at step **424** to store the authentication token and the timestamp T_0 in the client's state data structure. This prevents a reuse of the authentication cookie to ensure that the scheme is protected against a replay attack. This information is removed from the client's state information once the timestamp has expired.

10 Thus, upon receipt of the identity cookie and the request's associated authentication token, the server S determines the identity of the sender from the identity cookie and authenticates the request to access the protected resource against the authentication token. If
15 the authentication token is not valid, the server S returns to the client an access denied message and the protocol terminates. If, however, the authentication token is valid, then the protocol continues as follows.

20 At step **426**, the server uses the ACT_{CS} (which is found in the client state data structure) to make an access control decision with respect to the URL to which the client desires access. The authentication token is stored at the server S for the "lifetime" of the token (which will be quite short, as described above) to
25 enforce the one-time only use of this cookie. If the

client is allowed to invoke the request URL, the invocation proceeds as normal at step **428**. If the client is not allowed to invoke the requested URL, the server sends the client an access denied message at step **430** and
5 the protocol ends.

Thus, according to the present invention, a one-time only authentication token is used to authenticate each request to access a protected resource. This token is unique to a given request and includes a time value that
10 is used to protect against replay attacks. The authentication token is regenerated for each new request. It is protected by a key shared between the client and the server.

When a target server B is different from the server
15 that established the authentication (but is still within the same Internet domain), the target B has no knowledge about the client's key k_c and no state information. Therefore, the target server is unable to validate the authentication token. The solution to this potential
20 problem is provided by the URL_{CS} that is contained inside the identity cookie. Preferably, this URL points to a script at the server S that has, as input, the client's UID_{CS} , the URL_{VISIT} the client wants to access, and the authentication token. Execution of this script then
25 returns an answer to the question regarding whether the

authentication token is valid. **Figure 5** illustrates a flowchart of the process.

The routine begins at step **500** by examining the client's state using UID_{CS} . At step **502**, the routine
5 compares the received timestamp T_0 to the timestamp contained in the client's state information to prevent a replay attack. If the timestamps do not match, an access denied message is issued at step **503**. If the timestamps are the same, the routine continues at step **504** to
10 calculate the MAC. At step **506**, the routine compares the MAC to the MAC inside the authentication token. If the outcome of the test at step **506** indicates that the MACs do not match, an access denied message is generated at step **510**. If, however, the MACs match, the script sends
15 back a positive response at step **508** to complete the process. The target web server can then obtain the client's ACT_{CS} as it is convinced about the client's identity (namely, UID_{CS} , which, for example, is the client's DCE principal name). Note that if the identity
20 cookie contains the client's access control token, ACT_{CS} , and ACT_{CS} has been suitably protected, the target web server does not need to obtain this information from another source. If the ACT_{CS} is not included in the identity cookie then the target web server must call out
25 to an authorization server to determine the value of

ACT_c. Preferably, URL_{cs} is secured within an ACL and is contacted over an SSL/TLS protocol connection, which prevents an outsider from intercepting or changing any information.

5 The inventive protocol may also be implemented in a cross domain situation. In one embodiment, a target server B (in domain B) knows that S is the authentication server in domain S. This case occurs, for example, when the user wishes to access a protected resource that is
10 outside of its DNS domain. The target server (B) also has no knowledge of the user. Because the target server is out-of-domain, there needs to be a way to communicate this information to the target server from the initial authentication server (S). This may be achieved as
15 described in the flowchart of **Figure 6**.

 The routine begins at step **600** when the plug-in sends the request for the protected resource to the target server B. Because B is out-of-domain, the user's identity cookie is not forwarded to B. At step **602**, the
20 target server B redirects the request to the authentication server S. Because server B knows that server S is the authentication server, this information is established outside of the protocol. At this point, the user's identity cookie is forwarded to S by the
25 user's browser. At step **604**, the server S performs a test

to authenticate the client based on the authenticator token as previously described. If the request is valid (authentic, not necessarily authorized), the routine continues at step **606**. At this step, server S redirects
5 the request to server B with the value k_{BC} encrypted by a long-term shared key K_{SB} , where:

$$k_{BC} = h^{(s)}(k_{SC}, B)$$

K_{SB} is a long, term shared, symmetric key between server B and server S. Preferably, this key is communicated
10 between servers B and S through a back channel, e.g., a secure SSL/TLS communications channel. However this key is generated must be easily reproducible by the client without any intervention by server S. This allows the client to request protected resources at server B without
15 intervention by server S.

At step **608**, server S sends server B an "authentication okay" response that indicates to B that the request has been authenticated. At step **610**, server B creates an identity cookie for C at B, called IDC_{CB} ,
20 that may or may not contain the user's access control token in the domain of B, such that

$$IDC_{CB} = [UID_{CB}, B, URL_{CB}], (ACT_{CB})$$

At step **612**, puts the identity cookie in C's browser. The shared C-B key and the identity cookie mean that
25 future requests to server B need not invoke server S.

Once the user has been "set up" at the out-of-domain server B in this fashion, it can generate "normal" requests as if server B were in-domain, where an authentic request to access a protected resource in the domain of B is defined as

$$ANC_C = N_B, B, T_0, UID_{CB}, MAC(k_{BC})(N_B, B, T_0, UID_{CB}, URL_{VISIT}) .$$

If the outcome of the test at step **604** indicates that the request is not authenticated, the routine branches to step **614**. At this step, server S requires the client to re-login. Server S then redirects the required information to server B upon successful login. This completes the process.

The cross-domain protocol may be used if the client does not have an identity cookie at the target server B or if the client has already registered into domain B. In the former case, registration information must be transferred from the registering domain (server S) to the requested domain (server B). As described above, preferably this is accomplished using a secure, back channel. In the latter case, the protocol is the same as that illustrated in **Figure 4**, where the server S has simply been replaced by the server B.

The present invention provides numerous advantages. The system offers a web-based single sign-on to end users desiring access to protected resources within and across

authentication domains. The system enables the storage and maintenance of state information at the client side without having to install and configure specialized software (this is handled by the plug-in loaded by the signed applet). This state information preferably contains the ACT token which, in turn, enables use of a push model for an access control infrastructure. Further, as has been described, the protocol is not vulnerable to a replay attack. In particular, because the server keeps track of the previously used timestamp T_0 , it is not possible (except within a small window) to replay an authentication token. The protocol is lightweight, as the cryptographic functions used are high performance.

Although not described in detail, the protocol may be run over the SSL protocol to provide additional services such as data confidentiality.

The inventive protocol is implemented in a client-side piece of code and a server-side piece of code, as has been described. More generally, the inventive functionality is implemented in software executable in a processor, namely, as a set of instructions (program code) in a code module resident in the random access memory of the computer. Until required by the computer, the set of instructions may be stored in

another computer memory, for example, in a hard disk drive, or in a removable memory, or downloaded via the Internet or other computer network.

In addition, although the various methods described
5 are conveniently implemented in a general purpose computer selectively activated or reconfigured by software, one of ordinary skill in the art would also recognize that such methods may be carried out in hardware, in firmware, or in more specialized apparatus
10 constructed to perform the required method steps.

Having thus described our invention, what we claim as new and desire to secure by Letters Patent is set forth in the following claims.

15

CLAIMS

1. A method for determining whether to allow access to a protected resource from a server, comprising the steps of:

5 at a client, responsive to a request to retrieve the protected resource, generating a one-time only use piece of data which can be used to authenticate that the request is bound to a given identity contained in a cookie previously set by an authentication server;

10 forwarding the piece of data to the server in the request;

at the server, determining whether the piece of data is valid; and

15 if the piece of data is valid, executing an access control decision to determine whether to invoke the request.

2. The method as described in Claim 1 wherein the one-time only use piece of data is generated by applying
20 a given function to a URL of the protected resource, a timestamp, a nonce generated by a server, the server's identity, and the client's identity.

3. The method as described in Claim 2 wherein the
25 given function is a message authentication code (MAC)

calculated on the URL of the protected resource, the timestamp, the nonce, the server's identity, and the client's identity with a given key.

5 4. The method as described in Claim 3 wherein the given key is a symmetric key k_{sc} that binds the piece of data to the user identity contained in the identity cookie.

10 5. The method as described in Claim 4 wherein the symmetric key is generated by applying a one-way hash function to a shared client-server key k_c , the server identity, and a nonce.

15 6. The method as described in Claim 5 wherein the shared client-server key is generated by applying a one-way hash function to a user password.

20 7. The method as described in Claim 1 wherein the cookie includes a userid, the server identity, and a URL pointing to a location at the server that includes a script.

25 8. The method as described in Claim 1 wherein the cookie includes a userid, the server identity, and a URL

pointing to a location at the server that includes a script, and an access control token.

9. The method as described in Claim 8 wherein the
5 script includes code for identifying whether a given piece of data is valid.

10. The method as described in Claim 9 wherein the
script is accessed if the protected resource is located
10 on a server other than the authentication server and the server and the authentication server are located within the same authentication domain.

15

11. A method of accessing a protected resource at a server, comprising the steps of:

at the server, receiving a request for a URL together with an identity cookie and a one-time only use authentication token associated with the request;

determining whether the authentication token is valid;

if the authentication token is not valid, returning to a requesting client an access denied message; and

10 if the authentication token is valid, executing an access decision function to determine whether to allow access to the protected resource.

12. The method as described in Claim 11 wherein the authentication token comprises a message authentication code (MAC) calculated on a URL of the protected resource, a nonce generated by the server, the server's identity, a user's identity, and a timestamp with a given key.

20 13. The method as described in Claim 12 wherein the given key is a symmetric key k_{sc} that binds the piece of data to the user identity as defined in the identity cookie.

14. The method as described in Claim **11** wherein the identity cookie includes a userid, an optional access control token, and a URL pointing to a location that includes a script.

5

15. The method as described in Claim **12** wherein the step of determining whether the authentication token is valid includes the steps of:

calculating a message authentication code;

10 evaluating whether the message authentication code is the same as the MAC in the authentication token.

16. The method as described in Claim **12** further including the step of saving the timestamp in a data
15 structure to prevent replay of the authentication token.

17. The method as described in Claim **16** further including the step of saving a nonce generated by the server, the server's identity, and the user's identity to
20 prevent replay of the authentication token by a client other than the user.

18. A computer program product in a computer-useable medium executable by a processor in a client computer, comprising:

code, responsive to a request to a server for
5 retrieval of a protected resource, which generates a unforgeable piece of data which can be used at the server to authenticate that the request is bound to a given identity contained in a cookie previously set by an authentication server; and
10 code for inserting the piece of data into the request to the server.

19. The computer program product as described in Claim 18 further including a signed applet for installing
15 the code in the client computer.

20. The computer program product as described in Claim 18 wherein the code which generates the unforgeable piece of data comprises:
20 code for calculating a message authentication code (MAC) on a URL of the protected resource, a nonce generated by a server, the server's identity, a user's identity and a timestamp with a given key.

21. The computer program product as described in Claim 20 wherein the code for calculating the message authentication code further includes code for generating the given key.

5

22. The computer program product as described in Claim 21 wherein the given key is a symmetric key that binds the piece of data to the user's identity contained in the cookie.

10

11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200

23. A computer program product for use in a computer-useable medium executable by a processor in a server, comprising:

code responsive to receipt of a request for a URL
5 for a protected resource together with a one-time only use authentication token associated with the request for determining whether the authentication token is valid;

code for returning to a requesting client an access denied message if the authentication token is not valid;
10 and

code for controlling execution of an access decision function if the authentication token is valid.

24. The computer program product as described in
15 Claim 23 wherein the authentication token comprises a message authentication code (MAC) calculated on the URL of the protected resource, a nonce generated by a server, the server's identity, a user's identity, and a timestamp with a given key.

20

25. The computer program product as described in Claim 24 wherein the given key is a symmetric key k_{sc} that binds the piece of data to the user identity contained in an identity cookie set by an authentication server.

25

26. The computer program product as described in Claim 25 wherein the code for determining whether the authentication token is valid includes:

code for calculating a message authentication code;

5 and

code for evaluating whether the message authentication code is the same as the MAC in the authentication token.

10 27. The computer program product as described in Claim 23 further including code for saving the timestamp and the authentication token in a data structure to prevent replay of the authentication token.

15

28. A method for issuing an access request from a client browser to a server hosting a protected resource, wherein an identity cookie has been set on the client browser by an authentication server, comprising:

5 using a symmetric key to derive a message authentication code (MAC) on a URL of the protected resource and a timestamp;

 inserting the MAC together with the timestamp, the nonce set by the server, the server's identity, and a
10 user's identity into a header of the request; and

 forwarding the request to the server together with the identity cookie to enable the server to determine whether a requestor is authorized to access the protected resource.

15

29. The method as described in Claim 28 wherein a MAC is generated upon each request for the protected resource.

20 30. The method as described in Claim 28 wherein the symmetric key binds the MAC to a user identity contained in the identity cookie.

25 31. The method as described in Claim 30 wherein the symmetric key is generated by applying a one-way hash

function to a shared client-server key K_c , a nonce and the identity of the server that generated the nonce.

32. The method as described in Claim 31 wherein the
5 shared client-server key is generated by applying a one-way hash function to a user password.

33. The method as described in Claim 28 wherein the
identity cookie includes a userid, and a URL pointing to
10 a location that includes a script.

34. The method as described in Claim 28 wherein the
identity cookie includes a userid, a URL pointing to a
location that includes a script, and an access control
15 token.

35. The method as described in Claim 34 wherein the
script includes code for identifying whether a MAC is
valid.

20

**AUTHENTICATION AND AUTHORIZATION PROTOCOL FOR SECURE
WEB-BASED ACCESS TO A PROTECTED RESOURCE**

ABSTRACT OF THE DISCLOSURE

5 When a user makes a request to access a protected
resource identified by a URL, client-side code in a web
browser is used to generate an authentication token,
which is then sent to the server along with an identity
cookie that was set by that server. The authenticated
10 token is then used by the server to authenticate that the
request is properly tied to a given identity contained in
the identity cookie. If the authentication token can be
validated at the server, an access control decision is
then executed to determine whether to invoke the request
15 for the protected resource. If the authentication token
cannot be validated, an access denied request is returned
to the requesting client.

1/3
AUS990922US1
Hinton et al.

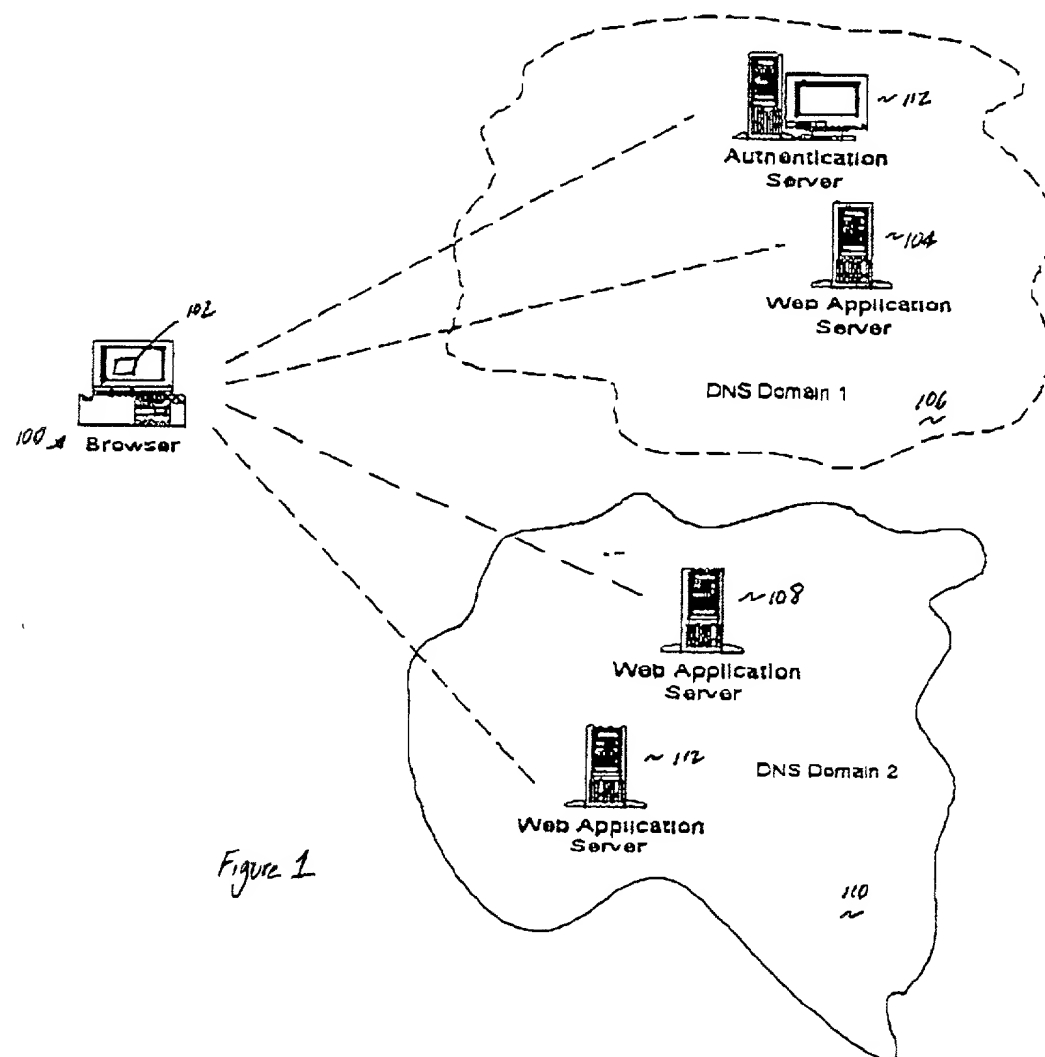


Figure 1

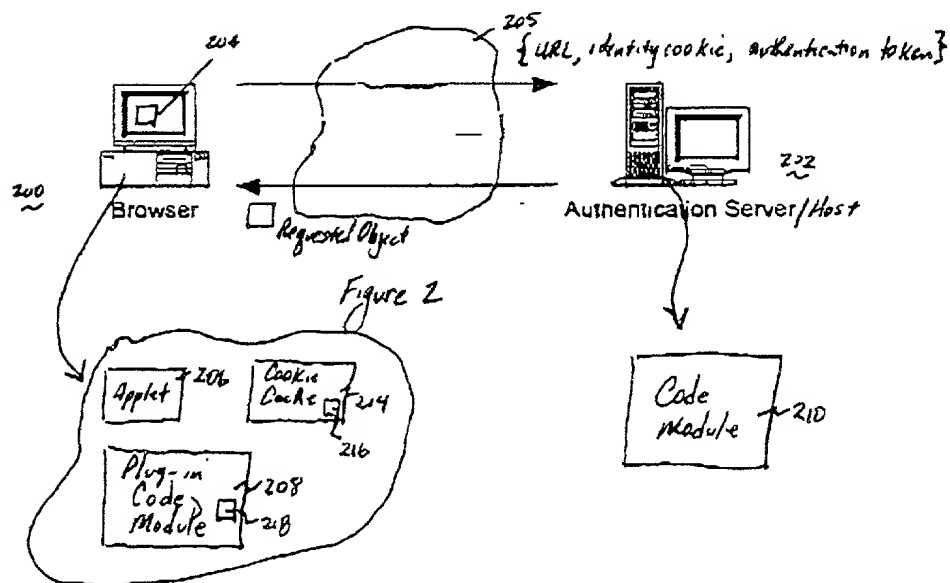


Figure 2

2/3
AUS990922US1
Hinton et al.

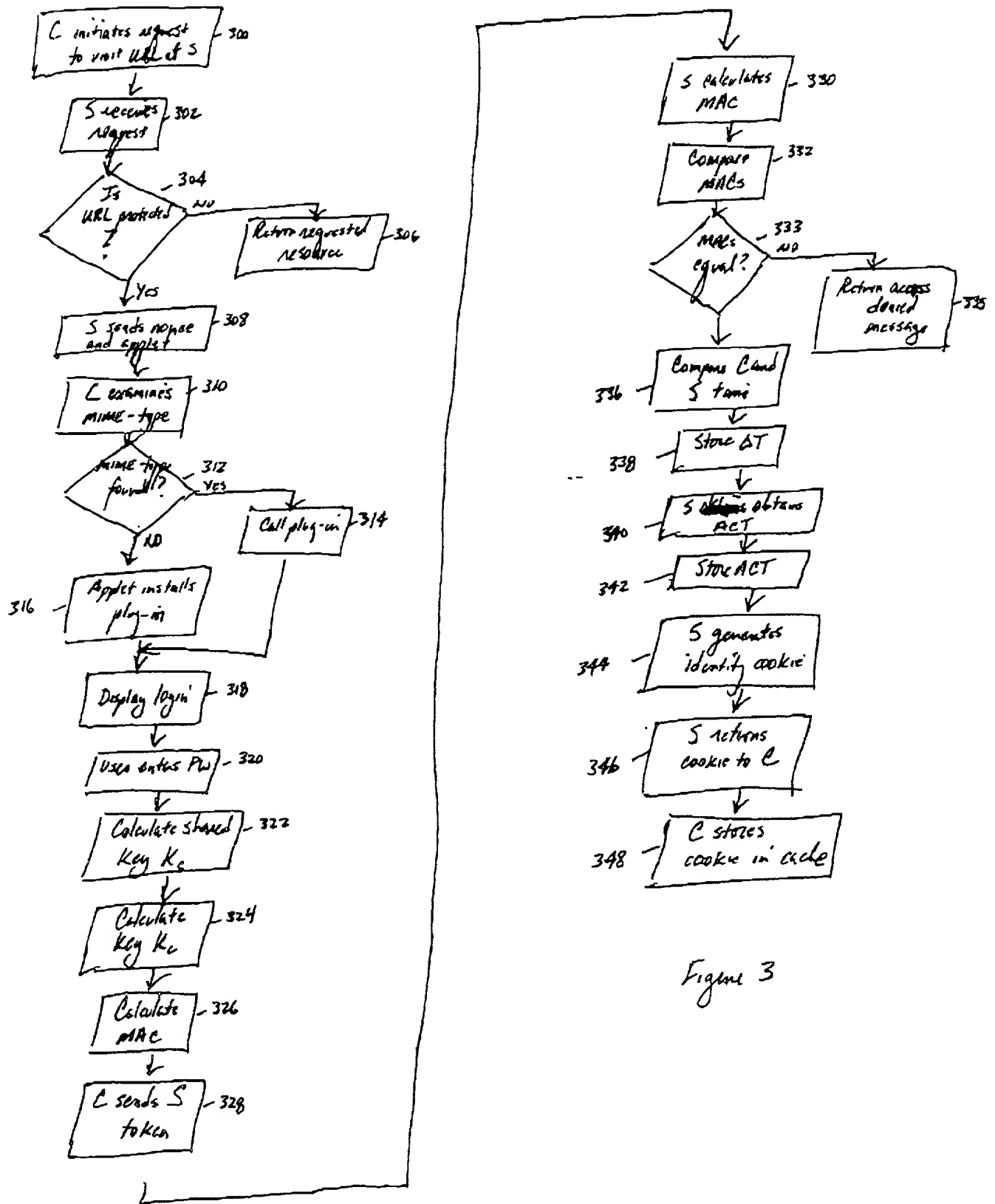


Figure 3

3/3
AUS990922US1
Hinton et al.

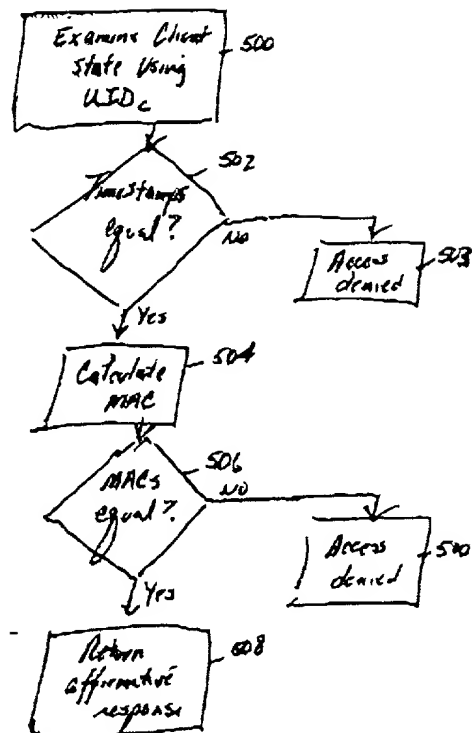
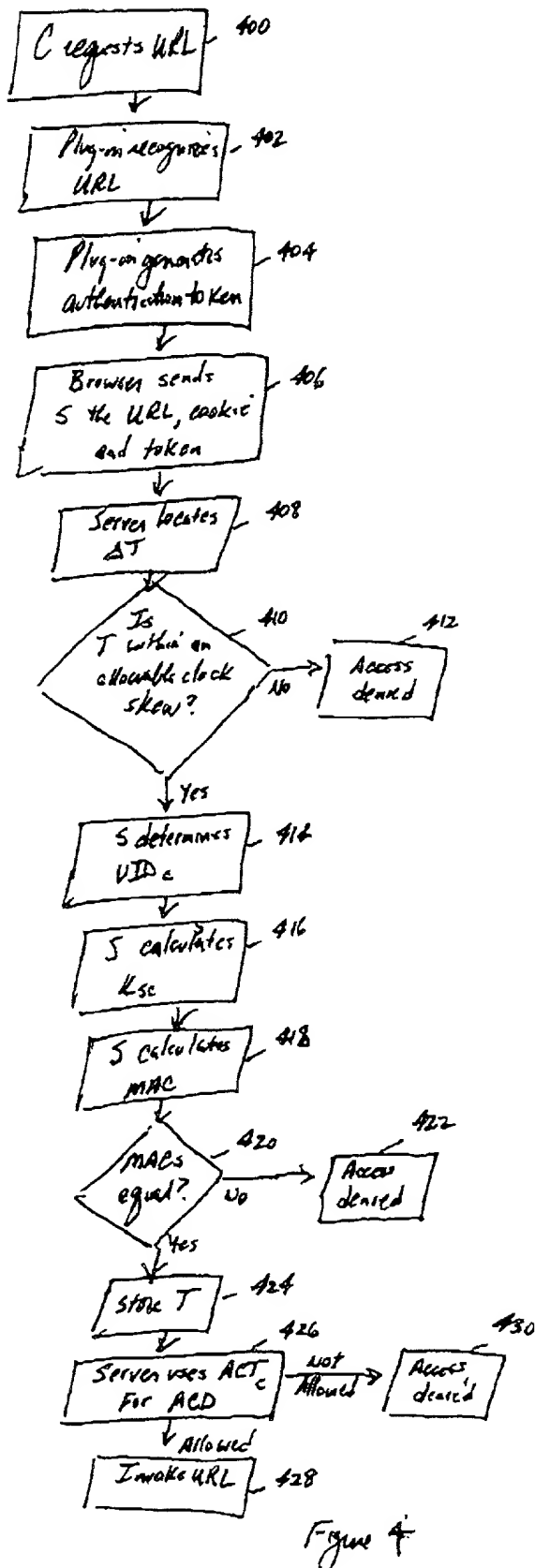


Figure 5

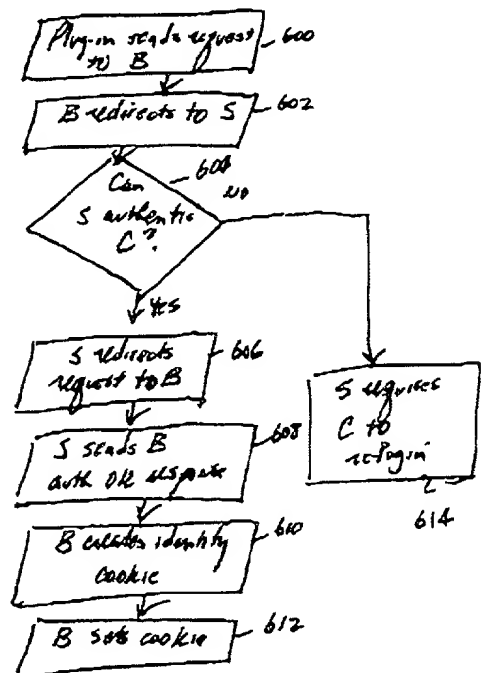


Figure 6

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

AUTHENTICATION AND AUTHORIZATION PROTOCOL FOR SECURE WEB-BASED ACCESS TO A PROTECTED RESOURCE

the specification of which (check one):

- ☒ is attached hereto.
- ☐ was filed on _____;
as Application Serial No. _____
and which was amended on _____ (if applicable)

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the patentability of this application in accordance with Title 37, Code of Federal Regulations, § 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, § 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s):

Priority Claimed

_____ (Number)	_____ (Country)	_____ (Day/Month/Year)	___ Yes	___ No
-------------------	--------------------	---------------------------	---------	--------

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose information material to the patentability of this application as defined in Title 37, Code of Federal Regulations, § 1.56 which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

(Application Serial #)

(Filing Date)

(Status)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.


POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorneys and/or agents to prosecute this application and transact all business in the Patent and Trademark Office connected therewith.

John W. Henderson, Jr., Reg. No. 26,907; James H. Barksdale, Jr., Reg. No. 24,091; Thomas E. Tyson, Reg. No. 28,543; Robert M. Carwell, Reg. No. 28,499; Jeffrey S. LaBaw, Reg. No. 31,633; Douglas H. Lefevre, Reg. No. 26,193; Casimer K. Salys, Reg. No. 28,900; David A. Mims, Jr., Reg. No. 32,708; Anthony V. England, Reg. No. 35,129; Volel Emile, Reg. No. 39,969; Leslie A. Van Leeuwen, Reg. No. 42,196; Christopher A. Hughes, Reg. No. 26,914; Edward A. Pennington, Reg. No. 32,588; John E. Hoel, Reg. No. 26,279; Joseph C. Redmond, Jr., Reg. No. 18,753; Marilyn S. Dawkins, Reg. No. 31,140; Mark E. McBurney, Reg. No. 33,114; David H. Judson, Reg. No. 30,467, and Douglas A. Sorensen, Reg. No. 31,570.

Send correspondence to: David H. Judson, Hughes & Luce, L.L.P., 1717 Main Street, Suite 2800, Dallas, Texas 75201 and direct all telephone calls to Mr. Judson at 214/9395672.

FULL NAME OF FIRST
INVENTOR:
INVENTOR'S SIGNATURE:

Heather Maria Hinton


31 MAY 2000

DATE:

RESIDENCE:

3512 Rip Ford Drive
Austin, Texas 78732
Canada

CITIZENSHIP:

FULL NAME OF SECOND
INVENTOR:
INVENTOR'S SIGNATURE:

Mark Vandenwauver

DATE:

RESIDENCE:

2601 Scofield Ridge Parkway #414
Austin, Texas 78727
Belgium

CITIZENSHIP:

5128383516

IBM DOCKET NO. AUS990922US1

**DECLARATION AND POWER OF ATTORNEY FOR
PATENT APPLICATION**

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

**AUTHENTICATION AND AUTHORIZATION PROTOCOL FOR SECURE
WEB-BASED ACCESS TO A PROTECTED RESOURCE**

the specification of which (check one):

- ☒ is attached hereto.
- ☐ was filed on _____;
as Application Serial No. _____
and which was amended on _____ (if applicable)

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the patentability of this application in accordance with Title 37, Code of Federal Regulations, § 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, § 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s):

Priority Claimed

(Number)	(Country)	(Day/Month/Year)	Yes	No
_____	_____	_____	___	___

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose information material to the patentability of this application as defined in Title 37, Code of Federal Regulations, § 1.56 which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

006372.00287:515821.01

5128383516

IBM DOCKET NO. AUS990922US1

(Application Serial #)

(Filing Date)

(Status)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorneys and/or agents to prosecute this application and transact all business in the Patent and Trademark Office connected therewith.

John W. Henderson, Jr., Reg. No. 26,907; James H. Barksdale, Jr., Reg. No. 24,091; Thomas E. Tyson, Reg. No. 28,543; Robert M. Carwell, Reg. No. 28,499; Jeffrey S. LaBaw, Reg. No. 31,633; Douglas H. Lefevre, Reg. No. 26,193; Casimer K. Salys, Reg. No. 28,900; David A. Mims, Jr., Reg. No. 32,708; Anthony V. England, Reg. No. 35,129; Volel Emile, Reg. No. 39,969; Leslie A. Van Leeuwen, Reg. No. 42,196; Christopher A. Hughes, Reg. No. 26,914; Edward A. Pennington, Reg. No. 32,588; John E. Hoel, Reg. No. 26,279; Joseph C. Redmond, Jr., Reg. No. 18,753; Marilyn S. Dawkins, Reg. No. 31,140; Mark E. McBurney, Reg. No. 33,114; David H. Judson, Reg. No. 30,467, and Douglas A. Sorensen, Reg. No. 31,570.

Send correspondence to: David H. Judson, Hughes & Luce, L.L.P., 1717 Main Street, Suite 2800, Dallas, Texas 75201 and direct all telephone calls to Mr. Judson at 214/9395672.

FULL NAME OF FIRST

Heather Maria Hinton

INVENTOR:

INVENTOR'S SIGNATURE:

DATE:

RESIDENCE:

3512 Rip Ford Drive
Austin, Texas 78732
Canada

CITIZENSHIP:

FULL NAME OF SECOND

Mark Vandenwauver

INVENTOR:

INVENTOR'S SIGNATURE:

DATE:

RESIDENCE:

2601 Scofield Ridge Parkway #414
Austin, Texas 78727
Belgium

CITIZENSHIP:

006372.00287;515821.01